OBJECT ORIENTED DATA BASE MANAGING DEVICE

Patent Number: JP7182179
Publication date: 1995-07-21
Inventor(s): TANAKA KEI
Applicant(s):: FUJI XEROX CO LTD
Requested Patent: JP7182179
Application Number: JP19930327759 19931224
Priority Number(s):
IPC Classification: G06F9/45 ; G06F9/44 ; G06F12/00 ; G06F17/30
EC Classification:
Equivalents:

Abstract

PURPOSE:To improve the processing efficiency for conversion of a class form by eliminating the dynamic analysis of the class type that is defined by a user.
CONSTITUTION:A compiler 10 compiles a source code 30, and the compiling result of the class type defined by a user is registered on a type symbol table 50 as the type information. A type structure analyzing part 23 generates and output a form converter means 60 concerning the defined class type based on the type information registered on the table 50 and the contents which are held at a basic data type form converting/holding part 21 and a form converting/naming rule holding part 22. Then a linker 70 links the means 60 to an object program 40 outputted by the compiler 10 and generates a practicable program 80.

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平7－182179

(43)公開日　平成７年(1995)７月21日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ	9/45					
	9/44	５３０　Ｐ	9193－5B			
	12/00	５４７　Ａ	8944－5B			
			9292－5B	Ｇ０６Ｆ　9/44	３２２　Ｚ	
			9194－5L	15/40	３８０　Ｅ	

審査請求　未請求　請求項の数１　ＯＬ　（全７頁）　最終頁に続く

(21)出願番号　特願平5－327759

(22)出願日　平成５年(1993)12月24日

(71)出願人　000005496
富士ゼロックス株式会社
東京都港区赤坂三丁目３番５号

(72)発明者　田中　圭
神奈川県川崎市高津区坂戸３丁目２番１号
ＫＳＰ　Ｒ&Ｄ　ビジネスパークビル
富士ゼロックス株式会社内

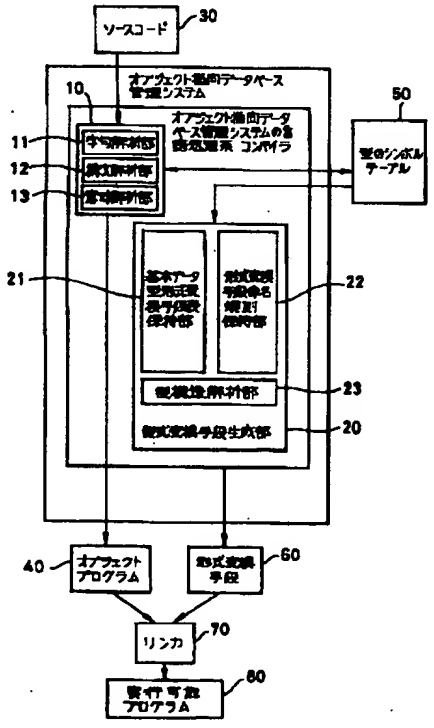
(74)代理人　弁理士　木村　高久

(54)【発明の名称】　オブジェクト指向データベース管理装置

(57)【要約】

【目的】ユーザ定義されたクラスの型の動的な解析を無くして、クラスの形式変換の処理効率を向上させることのできるオブジェクト指向データベース管理装置を提供する。

【構成】コンパイラ１０はソースコード３０をコンパイルする。ソースコード３０のうち、ユーザ定義されたクラスの型についてのコンパイル結果は、型情報として型のシンボルテーブル５０に登録される。型構造解析部２３は、型のシンボルテーブル５０内の型情報と、基本データ型形式変換表保持部２１及び形式変換命名規則保持部２２に保持されている保持内容とに基づいて、上記ユーザ定義されたクラスの型についての形式変換手段６０を生成して出力する。リンカ７０は、形式変換手段６０とコンパイラ１０によって出力されるオブジェクトプログラム４０とをリンクして、実行可能プログラム８０を生成する。



1

【特許請求の範囲】

【請求項1】二次記憶上の格納形式及び一次記憶上の計算可能形式の相互の形式変換を行うオブジェクト指向データベース管理装置において、

ユーザ定義されたクラスを含むソースコードのコンパイル時に、当該クラスの型の構造を解釈し、該解釈結果に対応じて、実行時に前記形式変換を行う形式変換手段を作成する作成手段を具備したことを特徴とするオブジェクト指向データベース管理装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、二次記憶上の格納形式及び一次記憶上の計算可能形式の相互の形式変換を行うオブジェクト指向データベース管理装置に関する。

【0002】

【従来の技術】従来、オブジェクト指向データベース管理システムにおいては、永続的インスタンスの主記憶上での計算可能形式と二次記憶上での格納形式とが異なっている場合であっても、永続的インスタンスを用いた処理の実行が可能なように、上記計算可能形式と上記格納形式との形式変換を行う形式変換手段を設けるようにしている。

【0003】なお上記形式変換手段を備えたシステムとしては、例えば特開平3－137730号公報（発明の名称：オブジェクト指向コンピュータ・システム）に記載されたものが知られている。この公報に記載されたものは、システムが永続的インスタンスの形式変換手段を有するクラスを提供し、そして、データベース管理システムの利用者（すなわちアプリケーションの作成者）によって、前記形式変換手段を有するクラスのサブクラスとして新たなクラスが定義されると、前記形式変換手段を有するクラスに基づいて、この新たに定義されたクラスの永続的インスタンスの形式を変換するようにしている。

【0004】すなわちオブジェクト指向のプログラミング技法には周知のように継承（インヘリタンス）の機能があるので、この継承機能を用いることにより、上記形式変換手段を有するクラスのサブクラスにも当該形式変換手段が継承されることになるので、当該形式変換手段を用いて新たに定義されたクラスの永続的インスタンスの形式を変換するようにしている。

【0005】

【発明が解決しようとする課題】ところで、一般的に、オブジェクト指向データベース管理システムにおいてはユーザが自由にクラスの型を定義できるようになっている。

【0006】しかし、上記公報のものでは、永続的インスタンスの形式変換は必ずシステムが予め用意した1種類の永続的なインスタンスの形式変換手段によって行うようにしているので、この形式変換手段によって、実行

2

可能プログラムの実行時に、ユーザにより定義された各々の永続的インスタンスの型を動的に解析しながら形式変換を実行しなければならない。そのため、予め用意された1種類の形式変換手段による形式変換の処理効率の向上は望めない。またプログラムの実行効率の向上も望めない。

【0007】本発明は、ユーザ定義されたクラスの型の動的な解析を無くして、クラスの形式変換の処理効率を向上させることのできるオブジェクト指向データベース管理装置を提供することを目的とする。

【0008】

【課題を解決するための手段】この発明は、二次記憶上の格納形式及び一次記憶上の計算可能形式の相互の形式変換を行うオブジェクト指向データベース管理装置において、ユーザ定義されたクラスを含むソースコードのコンパイル時に、当該クラスの型の構造を解釈し、該解釈結果に対応じて、実行時に前記形式変換を行う形式変換手段を作成する作成手段を具備している。

【0009】

【作用】この発明では、作成手段が、ユーザ定義されたクラスを含むソースコードのコンパイル時に、当該クラスの型の構造を解釈し、更にこの解釈結果である型に対応して、実行時に前記形式変換を行う形式変換手段を作成する。従って、実行時に、クラスの型の動的な解析を行うこと無く、形式変換手段によりクラスの形式変換を効率良く行うことができる。

【0010】

【実施例】以下、本発明の実施例を添付図面を参照して説明する。

【0011】図1は本発明に係るオブジェクト指向データベース管理装置を適用したオブジェクト指向データベース管理システムの一実施例の構成を示す機能ブロックである。

【0012】図1において、システムは、コンパイラ10と形式変換手段生成部20とを備えている。コンパイラ10はオブジェクト指向データベース管理システムの言語処理系であり、字句解析部11、構文解析部12及び意味解析部13を備え、これらの機能部により、ユーザにより記述されたソースコード30（すなわち応用プログラムのソースファイル）をコンパイルすなわち字句解析、構文解析及び意味解析等を実行する。また、これらの解析が行われた（コンパイルされた）コンパイルコード（すなわちオブジェクトプログラム）40を出力する。

【0013】なお、これらの解析を逐次実行しながら、型のシンボルテーブル（以下、型シンボルテーブルという）50や、変数のシンボルテーブル（図示せず）を作成する。型シンボルテーブル50には、ユーザ定義されたクラスの型の型名、及びその型の構造が所定形式で表現されている。すなわち、ユーザ定義されたクラスの型

に関する情報が、上記型名及び型の構造から構成される型情報として格納される。なお、ユーザ定義されたクラスはクラス定義と呼ぶことができるので、以下、必要に応じてそれらの名称を併記する場合もある。

【００１４】一方、形式変換手段生成部２０は、基本データ型形式変換手段表保持部２１、形式変換手段命名規則保持部２２及び型構造解析部２３を備える。

＊【００１５】基本データ型形式変換手段表保持部２１は、システムが提供する基本データ型の形式変換手段を集めた表（以下、基本データ型形式変換手段表という）を保持している。

【００１６】本実施例で用いる基本データ型形式変換手段表の一例を以下に示す。

『

データ型	ロード手続	ストア手続
整数型	INT-LOAD()	INT-STOR()
文字列型	CHAR-STAR-LOAD()	CHAR-star-STORE()

』

……基本データ型形式変換手段表１

ここで、

ＬＯＡＤ（）は格納形式から計算可能形式への変換

ＳＴＯＲＥ（）は計算可能形式から格納形式への変換

をそれぞれ意味している。なお『』内の記述が一例を示している（以下の説明においても同様とする）。また整数型及び文字列型それぞれに対応するロード手続及びストア手続が、形式変換の変換手段としての機能を果たしている。この変換手段は本発明に係る形式変換手段その※

※ものではなく、当該形式変換手段に含まれるものである。

【００１７】形式変換手段命名規則保持部２２はユーザ定義されたクラスの型の形式変換手段の命名規則を保持している。

【００１８】本実施例で用いる変換手段命名規則の一例を以下に示す。

『

ロード関数	クラス名-Load()
ストア関数	クラス名-Store()

』　……形式変換手段命名規則１

型構造解析部２３は、コンパイラ１０により作成される型シンボルテーブル５０を入力として読み込み、この型シンボルテーブル５０と、基本データ型形式変換手段表保持部２１および形式変換手段命名規則保持部２２の保持内容とに基づいて、ユーザ定義のクラスの型の構造である型情報を解析し、ユーザ定義されたクラスの型に応じた最適な形式変換手段６０を作成し提供する。

【００１９】そして、オブジェクト指向データベース管理システムとは独立して設けられ、従来と同様の機能を果たすリンカ７０は、オブジェクトプログラム４０と形式変換手段６０とをリンクして、実行可能プログラム８０を生成する。なおこのリンカ７０は独立して設けることなく、オブジェクト指向データベース管理システム内に設けるようにしても良い。

【００２０】なお図１において、ソースコード３０、オブジェクトプログラム４０、型シンボルテーブル５０、形式変換手段６０、リンカ７０及び実行可能プログラム８０は共に、主メモリ（一次記憶上）に記憶され、必要に応じて外部記憶装置（二次記憶上）に格納される。またコンパイラ１０および形式変換手段生成部２０はそれぞれの機能を果たすソフトウェア（プログラム）を、中央処理装置等の制御手段が実行することにより実現される。勿論、ハードウェアやファームウェアで実現するようにしても良い。

【００２１】次に、本実施例のクラス定義の形式変換手★段の生成処理について図２のフローチャートを参照して説明する。

【００２２】なお、クラスの型の形式変換は読み込み（ロード）と書き込み（ストア）の２種類について行われるので、ロード関数とストア関数の２種類の形式変換手段が必要となる。ここでは、ロード関数の形式変換手段の生成について説明するが、ストア関数の場合も同じ手順で生成することができる。またここでは、Ｃ言語系すなわち「Ｃ＋＋言語」を例にとって説明する。

【００２３】最初に、コンパイラ１０は、ユーザ定義されたクラス（すなわちクラス定義）を含むソースコード３０をコンパイルしながら（ステップ１１０）、型シンボルテーブル５０を生成する（ステップ１２０）。またコンパイル処理を終了した場合は、オブジェクトプログラム４０を出力する。

【００２４】ここで、「Ｃ＋＋言語」による記述によりユーザ定義されたクラスの一例を説明するが、それに先立って、「Ｃ＋＋言語」におけるクラスの型の表現形式を以下に示す。

5　　6

```
クラス　クラス名｛
属性型　属性名

｝；
```

次にクラス定義の一例を以下に示す。

```
class Foo {
    int a;
    char* b;
    class Bar* c;
};
```

……クラス定義（ユーザ定義されたクラス）1

ここで、クラス定義1においては3つの属性が存在している。

【0025】なおクラス定義1を詳細に説明すると以下の様になっている。

◇「class 」はクラス、「Foo 」はクラス名を表している。

◇「int a;」は第1番目の属性であり、「int 」は属性型（ここでは整数型）を表し、「a 」は属性名を表している。

◇「char* b;」は第2番目の属性であり、「char* 」は属性型（ここでは文字型へのポインタ）を表し、「b 」は属性名を表している。

◇「class Bar* c; 」は第3番目の属性であり、「class Bar*」は属性型（ここではバー型オブジェクトへのポインタ）を表し、「c 」は属性名を表している。

また、型シンボルテーブル50に格納される型情報は、上記クラス定義1がコンパイラ10によってコンパイルされた結果となっている。

【0026】上述した様にして型シンボルテーブル50が作成されたならば、型構造解析部23は、その型シンボルテーブル50を入力として読み込み（ステップ130）、型シンボルテーブル50に登録されている型情報内のクラス名と、形式変換手段命名規則保持部22に保持されている上記変換手段命名規則1とに基づいて、ユーザ定義されたクラスの形式変換手段名を決定する（ステップ140）。

【0027】この実施例では、上記型情報は上記クラス定義1に対応する内容なので、上記型情報のクラス名は「Ｆｏｏ」であることが分かる。従ってこのクラス名「Ｆｏｏ」と上記形式変換手段命名規則1内の「ロード関数　クラス名-Load() 」規則を用いて、ユーザ定義されたクラスの形式変換手段を「Ｆｏｏ－Ｌｏａｄ（）」と決定する。

【0028】次に、型構造解析部23は、型シンボルテーブル50内の型情報のうち、ユーザ定義のクラスに関＊するものを取り出し、そのクラスの属性の型（すなわち属性型）は存在しているか否かを判断し（ステップ150）、属性が存在している場合は、属性は基本データ型か否かを判断する（ステップ160）。

【0029】ここで、基本データ型の場合は、上記基本データ型形式変換手段表1から適切な変換手段を選択し、この変換手段を、このユーザ定義されたクラスの形式変換手段から呼び出される手続きとする（ステップ170）。このステップが終了した後は、上記ステップ150に移行する。

【0030】一方、ステップ160において基本データ型でない場合は、上記形式変換命名規則1を用いて求めた適切な変換手段を、このユーザ定義されたクラスの形式変換手段から呼び出される手続きとする（ステップ180）。このステップが終了した後は、上記ステップ150に移行する。

【0031】すなわち、型構造解析部23は、型シンボルテーブル50内の型情報のうち、ユーザ定義されたクラスに関するものを取り出し、そのクラスの属性の型（つまり属性型）について逐次解析することによって、そのクラスの型の構造を解析する。このクラスの型の構造の解析結果を元に、各属性毎に、適切な変換手段を決定し、これを組合わせることにより、そのクラスの形式変換手段を生成する。これら各属性の適切な変換手段の決定は、その属性が基本データ型であれば基本データ型の形式変換を集めた表を引くことにより決定でき、ユーザ定義のクラスであれば、上記形式変換手段命名規則1を適用して決定できる。

【0032】ここで、ステップ150～180の処理について具体例を挙げて説明する。この実施例においては、基本データ型には、上記基本データ型形式変換手段表1に示しているように、「ＩＮＴ」（整数型）と「ＣＨＡＲ」へのポインタ（文字列型）の2つが含まれている。一方、上記クラス定義1には、上述した説明から明らかな様に3つの属性が含まれている。そこで次に、こ

れらの属性について基本データ型かどうかをチェックする。

【００３３】第１番目の属性は「ＩＮＴ」なので整数型の基本データ形式であることが分かるので、ステップ１５０およびステップ１６０を経てステップ１７０が実行される。すなわち、ステップ１７０においては、上記基本データ型形式変換手段表１から、ロード手続であり整数型である「ＩＮＴ－ＬＯＡＤ（）」の変換手段を選択し、この変換手段「ＩＮＴ－ＬＯＡＤ（）」を、上記形式変換手段「Ｆｏｏ－Ｌｏａｄ（）」で最初に呼び出される手続きとする。

【００３４】また第２番目の属性は「ＣＨＡＲ」へのポインタなので文字列型の基本データ形式であることが分かるので、上記同様に、基本データ型形式変換手段表１から、ロード手続であり文字列型である「ＣＨＡＲ－ＳＴＡＲ－ＬＯＡＤ（）」の変換手段を選択し、この変換手段「ＣＨＡＲ－ＳＴＡＲ－ＬＯＡＤ（）」を、上記形式変換手段「Ｆｏｏ－Ｌｏａｄ（）」で２番目に呼び出される手続きとする。

【００３５】そして、第３番目の属性については、「ｃ＊ｌａｓｓ　Ｂａｒ＊　ｃ」であり、上記基本データ型形式変換手段表１には存在していないないので、基本データ型ではない、すなわちユーザ定義されたクラスの型ということが分かり、ステップ１５０及びステップ１６０を経てステップ１８０が実行される。このステップ１８０においては、現在の変換処理についてはロード関数であるという点と、上記形式変換手段命名規則１とに基づいて、「クラス名－Ｌｏａｄ（）」が得られ、よって、適切な変換手段を「Ｂａｒ－Ｌｏａｄ（）』と決定し、この変換手段を上記形式変換手段「Ｆｏｏ－Ｌｏａｄ（）」で３番目に呼び出される手続きとする。

【００３６】以上の処理で、ユーザ定義されたクラスの３つの属性についての形式変換手段の決定が終了したことになるので、次に、型構造解析部２３は、これらの形式変換手段により得られた上記各変換手段を組合わせることにより、ユーザ定義されたクラスの形式変換手段を生成する。

【００３７】このようにして得られたロード関数についての形式変換手段の一例を以下に示す。なお、形式変換手段の表現形式は以下のようになっている。

『
C++言語又はC言語の関数定義
返値　関数名｛
手続き（関数で実行する手続き等）或いは文
｝
』

続いて、一例を以下に示す。

『
```
void Foo-Load() {
    INT-LOAD();
    CHAR-STAR-LOAD();
    Bar-Load();
}
```
』　…形式変換手段１

なお「ｖｏｉｄ」は返値無しを意味している。

【００３８】またストア関数についても上記図２に示されている処理手順で変換すると、そのストア関数についての形式変換手段は以下のようになる。

『
```
void Foo-Store() {
    INT-STORE();
    CHAR-STAR-STORE();
    Bar-Store();
}
```
』　…形式変換手段２

従って、上記形式変換手段１と形式変換手段２とを組合わせた以下のような形式変換手段が、上記クラス定義１に対応する形式変換手段すなわち形式変換手段６０となる。

【００３９】

『
```
void Foo-Load() {
    INT-LOAD();
    CHAR-STAR-LOAD();
```

```
    Bar-Load();
}
void Foo-Store() {
    INT-STORE();
    CHAR-STAR-STORE();
    Bar-Store();
}
```

なおこの形式変換手段６０は図１に示す形式変換手段６０を意味する。

【００４０】そして、上述したような形式変化手段６０とコンパイラ１０から出力されたオブジェクトプログラム４０とが、リンカ７０によってリンクされて、実行可能プログラム８０が生成される。

【００４１】以上説明したように本実施例によれば、コンパイル時において、ユーザ定義されたクラスの型に応じて、二次記憶上の格納形式と一次記憶上の計算可能形式との形式変換手段の生成を行うことができる。これは、実行可能プログラムの実行前に、ユーザ定義されたクラスの型に応じて、二次記憶上の格納形式と一次記憶上の計算可能形式との形式変換を行うための型の構造解析ができるということを意味している。

【００４２】従って、実行可能プログラムの実行時には、ユーザ定義されたクラスの型を動的に解析する必要がないので、実行可能プログラムの実行効率を向上させることができる。

【００４３】最後に、請求項１の発明の構成要件と図１に示した実施例の構成要素との対応関係について説明する。上記作成手段は図１に示した形式変換手段生成部２０及び型シンボルテーブル５０に対応している。

【００４４】

【発明の効果】以上詳細に説明したように本発明によれば、作成手段が、ユーザ定義されたクラスを含むソースコードのコンパイル時に、当該クラスの型の構造を解釈し、更にこの解釈結果である型に対応して、実行時に前記形式変換を行う形式変換手段を作成するようにしてい＊るので、コンパイル時において、ユーザ定義されたクラスの型に応じて、二次記憶上の格納形式と一次記憶上の計算可能形式との形式変換を行うための型の構造解析を行うことができる。すなわち、これは、実行可能プログラムの実行時に、ユーザ定義されたクラスの型に応じて、形式変換手段により、当該型の動的な解析を行うことなく、二次記憶上の格納形式と一次記憶上の計算可能形式との形式変換を行うことができるということを意味している。よって、従来と比較して、ユーザ定義されたクラスの型を動的に解析することなく、静的に解析することができる。また実行可能プログラムの実行効率を向上させることができる。

【００４５】従って、ユーザ定義されたクラスの型の動的な解析を無くして、クラスの形式変換の処理効率を向上させるオブジェクト指向データベース管理装置を提供することができるという利点がある。

【図面の簡単な説明】

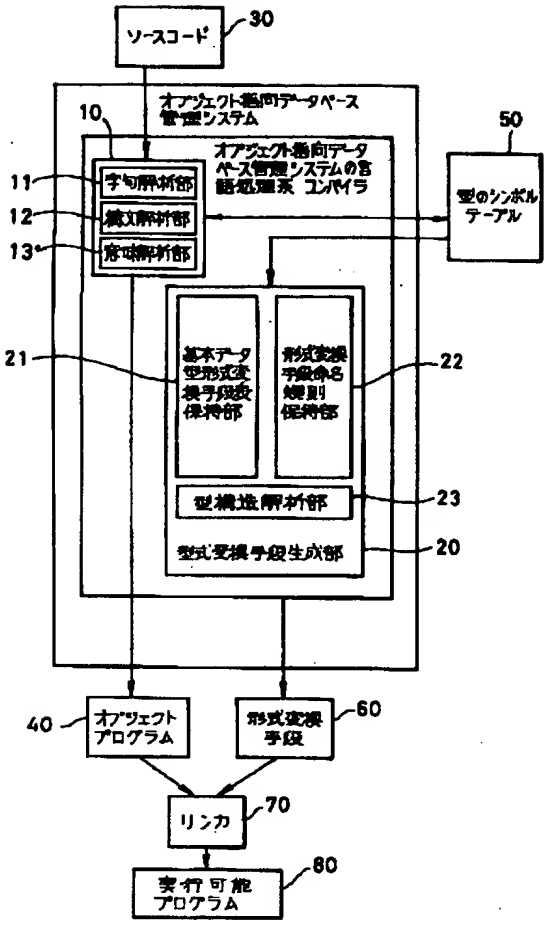
【図１】本発明に係るオブジェクト指向データベース管理装置の一実施例を示す機能ブロック図。

【図２】本実施例の形式変換手段の生成処理動作を示すフローチャート。

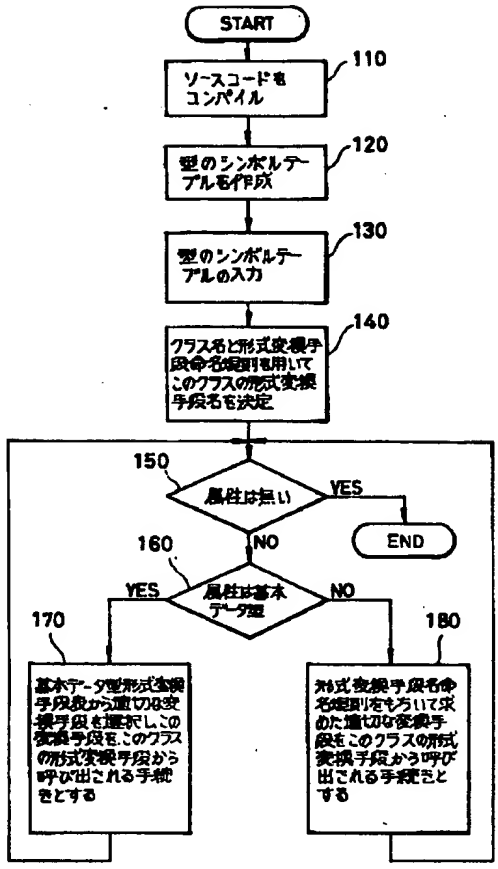
【符号の説明】

１０…ソースコード、１０…コンパイラ、２０…形式変換手段生成部、２１…基本データ型形式変換手段表保持部、２２…形式変換手段命名規則保持部２２、２３…型構造解析部、５０…型のシンボルテーブル、６０…形式変換手段。

【図1】



【図2】



フロントページの続き

(51)Int.Cl.⁶	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　17/30				